# Available Bandwidth Metrics for Application-Layer Reliable Multicast in Global Multi-Gigabit Networks

Kirill Karpov[1,2], Maksim Iushchenko[1,2], Nikolai Mareev[1,2], Dmytro Syzov[1], Eduard Siemens[1] and Viatcheslav Shuvalov[2]

[1] *Future Internet Lab Anhalt, Anhalt University of Applied Sciences, Bernburger Str. 57, Köthen, Germany*

[2] *Department of Transmission of Discrete Data and Metrology, Siberian State University of Telecommunications and Information Sciences, Kirova Str. 86, Novosibirsk, Russia*

{*kirill.karpov, maksim.iushchenko, nikolai.mareev, dmytro.syzov, eduard.siemens*}*@hs-anhalt.de, shvp04@mail.ru*

Keywords: Application Layer Multicast, Point-to-Multipoint, Available Bandwidth, Minimum Spanning Tree, Networking, High Bandwidth.

Abstract: Application layer multicast shows its efficiency when it is necessary to transmit enormous amount of data to many nodes. One of the most important issues for such kind of transmission is "what is the criteria for path construction?". In the global networks, when the distance between nodes becomes a significant factor, the time delay between nodes seems self-evident. However, in the presence of cross-traffic in the channel, the minimum spanning tree based on the delay might construct the tree which provides the lower data rate than possible alternative. The point of this work is to study how available bandwidth estimation techniques might solve such kind of challenges, how to adopt available bandwidth as a metric to construct data distribution paths and the cases when it gives higher performance in comparison with delay as a metric.

## 1 INTRODUCTION

In the current time, there is a possibility to create a global high bandwidth network infrastructure using cloud services for a relatively low cost. Cloud services provide the possibility to anyone to deploy own intercontinental network on top of the Internet or create own Content Delivery Network (CDN). However, dealing with the global network, it is necessary to deal with its properties, such as the high delay between nodes, the possibility of packet loss, interfering traffic (so-called cross-traffic) in the links, forbidden network features such as multicast. To make high-speed data transmissions over WAN possible, the RMDT [1] transport protocol was used, since it can overcome challenges described above: it protocol provides WAN acceleration service, which makes network packet losses and latency up to 1 second nearly negligible. It can serve up to 10 receivers in parallel within a single session without fairness issues meaning that available bandwidth will be shared evenly. It has a centralized congestion control, which allows the coexistence with the cross-traffic in IP WANs. The multicast functionality has been implemented and studied during the previous work [2] using the Minimal Spanning Tree (MST) algorithm. Previous work shows the efficiency of RMDT in conjunction with the MST algorithm using delay or RTT values between hosts as a metric in the global network. However, the presence of the cross-traffic in the global links is more than possible. Nevertheless that with the delay metric, the MST algorithm can construct the optimal tree, cross-traffic can negatively affect the bandwidth along a constructed path. This means that cross-traffic as a significant factor should be taken into account.

This paper studies the performance of application layer multicast in combination with high-bandwidth data transport applications using available bandwidth (AvB) metric and where and how it can outperform the delay metric.

The remainder of this paper is structured as follows. Section 2 provides an overview of related researches and methods. Section 3 gives the AvB metric description, how it can be obtained and adopted to the MST algorithm. Section 4 reviews the testing environment and software equipment that was used for the experimental setup. Section 5 is devoted to the retrieved results of experiments with application layer multicast implementation using RTT and AvB metrics. Interpretation and discussion of the results and future work can be found in Section 6.

## 2 RELATED WORK

An analysis of application layer multicast in the Wide Area Network has been made in the previous work [2]. It shows the efficiency of ALM in conjunction with Reliable Multi-Destination Transport Protocol (RMDT) in the global networks between hosts spaced across continents. In the research, the delay was used as a metric for the tree-first application layer multicast. The system has been tested in the AWS infrastructure.

The available bandwidth estimation research has been provided by Kirova, et. al [3, 4]. These researches introduce the Kite2 application for available bandwidth estimation, which is used in the given work.

The ALM model as a service implemented on the top of Hierarchical Peer to Peer Architecture, in order to give media streaming based applications or conferencing applications [5].

A multicast framework for point-to-multipoint and multipoint-to-point-to-multipoint video streaming from drones presented in the work [6]. The proposed rate-adaptive approach outperforms legacy multicast in terms of goodput, delay, and packet loss.

Adaptable, ISP-friendly multicast overlay network proposed in [7]. The system is adaptable to different conditions and easily reconfigurable in the construction and management of the multicast overlay distribution tree. The ALM tested has been tested in a network emulation tool.

There is an ALM based on an encoding-free non-dominated sorting genetic algorithm (EF-NSGA) [8]. The approach aimed to construct a tree based on multi-objective criteria: minimization transmission delay and instability simultaneously.

## 3 METRIC DESCRIPTION

The used AvB metric is based on Probing Rate Model (*PRM*) of the active probing measurement and uses analysis of inter-packet interval deviation on the receiver side (*IpIr*) in order to detect whether the sending rate of probe packets meets the available bandwidth limits of the link between the sender and the receiver.

Figure 1 illustrates the active probing model. $S_n$ is an $n$-th probe packet, $i_{sn}$ is the sending inter-packet interval and $i_{rn}$ is the inter-packet interval on the receiver side. The main principle of this method is to find the inter-packet interval, which is dependent on timing operations precision. The accurately chosen inter-packet interval on the sender side allows achieving the actual available bandwidth of the network path. The goal of the AvB algorithm is to find the minimal value $i_s$ after which the difference $i_r - i_s$ is minimal.

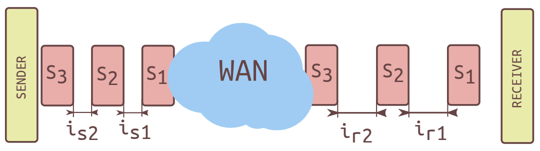


Figure 1: AvB active probing model [4].

Since the given ALM system uses the tree-first approach, the available bandwidth estimation values are collected only once as a first step. Collected metrics are formed into adjacency matrix A and inverted by Hadamard inverse rule as $A^{\circ -1}$ because the MST algorithm calculates the shortest tree where the minimal weight of an edge is preferable. The $A^{\circ -1}$ matrix passed to the MST or DCMST algorithm then. The resulted tree is treated as optimal from the perspective of the available bandwidth of the link.

## 4 EXPERIMENTAL SETUP

The section provides a detailed description of the testing infrastructure and software, which is used during the experiments.

### 4.1 Testing Environment

As an experimental environment, *Amazon AWS* has been chosen. It provides the virtual infrastructure in selected continents and regions. Cascade network transmission infrastructure based on **c5.xlarge** virtual instances, the configuration provided in Table 1.

Table 1: c5.xlarge host configuration.

| Name | Parameters |
|---|---|
| Operating system | Ubuntu 18.04 |
| CPU model | Intel Xeon Platinum 8000 (Cascade Lake) |
| CPU Frequency | 3.6 GHz |
| Number of vCPUs | 4 |
| RAM | 8 GB |
| Bandwidth | up to 10 Gbps |

The instances are distributed all over the world in the AWS regions, which shown in Figure 2. The regions are highlighted by colors here and further. **US West (Oregon)** - orange, **EU (Frankfurt)** - blue, **EU (London)** - red, **Asia Pacific (Singapore)** - cyan, **Canada (Central)** - green.

In each region, three **c5.xlarge** virtual instances have been deployed. The regions have been chosen to get different variations of network conditions, such as long and short distances, international and intercontinental links. The RTT probing results shown in Figure 3. The index rows and columns of the matrix are rep-resented each host by the first letter of its region and the number of the virtual machine in the region. Left indexes are output nodes, top indexes are input nodes. The matrix colored as a heatmap, where the cells with the highest values have more saturated color.

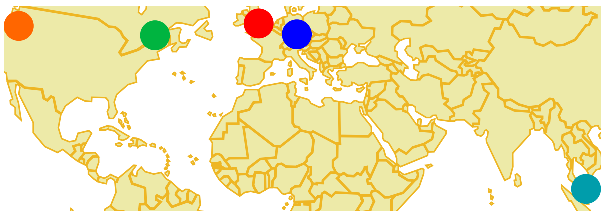
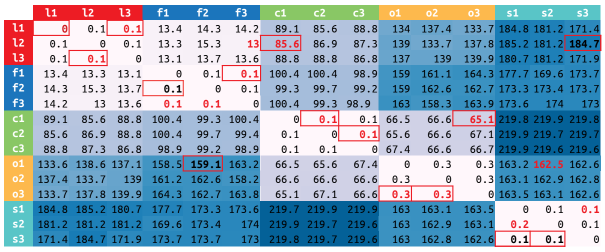


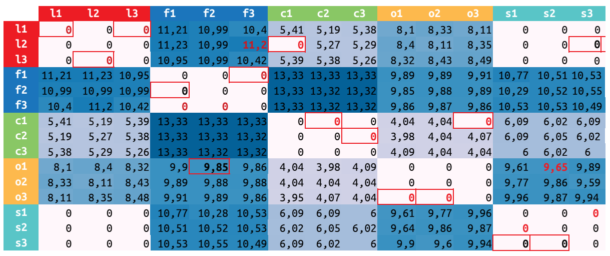
Figure 2: The map of AWS hosts location.



| | l1 | l2 | l3 | f1 | f2 | f3 | c1 | c2 | c3 | o1 | o2 | o3 | s1 | s2 | s3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| l1 | 0 | 0.1 | 0.1 | 13.4 | 14.3 | 14.2 | 89.1 | 85.6 | 88.8 | 134 | 137.4 | 133.7 | 184.8 | 181.2 | 171.4 |
| l2 | 0.1 | 0 | 0.1 | 13.3 | 15.3 | 13 | 85.6 | 86.9 | 87.3 | 139 | 133.7 | 137.8 | 185.2 | 181.2 | **184.7** |
| l3 | 0.1 | 0.1 | 0 | 13.1 | 13.7 | 13.6 | 88.8 | 88.8 | 86.8 | 137 | 139 | 139.9 | 180.7 | 181.2 | 171.9 |
| f1 | 13.4 | 13.3 | 13.1 | 0 | 0.1 | 0.1 | 100.4 | 100.4 | 98.9 | 159 | 161.1 | 164.3 | 177.7 | 169.6 | 173.7 |
| f2 | 14.3 | 15.3 | 13.7 | **0.1** | 0 | 0.1 | 99.3 | 99.7 | 99.2 | 159 | 162.6 | 162.7 | 173.3 | 173.4 | 173.7 |
| f3 | 14.2 | 13 | 13.6 | 0.1 | 0.1 | 0 | 100.4 | 99.3 | 98.9 | 163 | 158.3 | 163.9 | 173.6 | 174 | 173 |
| c1 | 89.1 | 85.6 | 88.8 | 100.4 | 99.3 | 100.4 | 0 | 0.1 | 0.1 | 66.5 | 66.6 | 65.1 | 219.8 | 219.9 | 219.8 |
| c2 | 85.6 | 86.9 | 88.8 | 100.4 | 99.7 | 99.4 | 0.1 | 0 | 0.1 | 65.6 | 66.6 | 67.1 | 219.9 | 219.7 | 219.7 |
| c3 | 88.8 | 87.3 | 86.8 | 98.9 | 99.2 | 98.9 | 0.1 | 0.1 | 0 | 67.4 | 66.6 | 66.7 | 219.9 | 219.6 | 219.6 |
| o1 | 133.6 | 138.6 | 137.1 | 158.5 | **159.1** | 163.2 | 66.5 | 65.6 | 67.4 | 0 | 0.3 | 0.3 | 163.2 | 162.5 | 162.6 |
| o2 | 137.4 | 133.7 | 139 | 161.2 | 162.6 | 158.2 | 66.6 | 66.6 | 66.6 | 0.3 | 0 | 0.3 | 163.1 | 162.9 | 162.8 |
| o3 | 133.7 | 137.8 | 139.9 | 164.3 | 162.7 | 163.8 | 65.1 | 67.1 | 66.6 | 0.3 | 0.3 | 0 | 163.5 | 163.1 | 162.6 |
| s1 | 184.8 | 185.2 | 180.7 | 177.7 | 173.3 | 173.6 | 219.7 | 219.9 | 219.9 | 163 | 163.1 | 163.5 | 0 | 0.1 | 0.1 |
| s2 | 181.2 | 181.2 | 181.2 | 169.6 | 173.4 | 174 | 219.9 | 219.7 | 219.6 | 163 | 162.9 | 163.1 | 0.2 | 0 | 0.1 |
| s3 | 171.4 | 184.7 | 171.9 | 173.7 | 173.7 | 173 | 219.8 | 219.7 | 219.6 | 163 | 162.8 | 162.6 | **0.1** | **0.1** | 0 |

Figure 3: Matrix of RTT metric values between hosts in milliseconds.

The experiments with additional cross-traffic are provided with the network load which can be seen in the matrix in Figure 4. The matrix is generated in the way to let MST algorithm construct the path, based on the AvB metric, which is similar to the RTT-based tree, but with two swapped branches: *frankfurt* and *singapore* hosts. Such configuration aimed to confuse RTT probing and tree construction process, which re-turn the non-optimal tree from the AvB perspective.



| | l1 | l2 | l3 | f1 | f2 | f3 | c1 | c2 | c3 | o1 | o2 | o3 | s1 | s2 | s3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| l1 | 0 | 0 | 0 | 11,21 | 10,99 | 10,4 | 5,41 | 5,19 | 5,38 | 8,1 | 8,33 | 8,11 | 0 | 0 | 0 |
| l2 | 0 | 0 | 0 | 11,23 | 10,99 | 11,2 | 0 | 5,27 | 5,29 | 8,4 | 8,11 | 8,35 | 0 | 0 | **0** |
| l3 | 0 | 0 | 0 | 10,95 | 10,99 | 10,42 | 5,39 | 5,38 | 5,26 | 8,32 | 8,43 | 8,49 | 0 | 0 | 0 |
| f1 | 11,21 | 11,23 | 10,95 | 0 | 0 | 0 | 13,33 | 13,33 | 13,33 | 9,89 | 9,89 | 9,91 | 10,77 | 10,51 | 10,53 |
| f2 | 10,99 | 10,99 | 10,99 | **0** | 0 | 0 | 13,33 | 13,32 | 13,32 | 9,85 | 9,88 | 9,89 | 10,29 | 10,52 | 10,55 |
| f3 | 10,4 | 11,2 | 10,42 | 0 | 0 | 0 | 13,33 | 13,32 | 13,32 | 9,86 | 9,87 | 9,86 | 10,53 | 10,53 | 10,49 |
| c1 | 5,41 | 5,19 | 5,39 | 13,33 | 13,33 | 13,33 | 0 | 0 | 0 | 4,04 | 4,04 | 0 | 6,09 | 6,02 | 6,09 |
| c2 | 5,19 | 5,27 | 5,38 | 13,33 | 13,33 | 13,32 | 0 | 0 | 0 | 3,98 | 4,04 | 4,07 | 6,09 | 6,05 | 6,02 |
| c3 | 5,38 | 5,29 | 5,26 | 13,33 | 13,32 | 13,32 | 0 | 0 | 0 | 4,09 | 4,04 | 4,04 | 6 | 6,02 | 6 |
| o1 | 8,1 | 8,4 | 8,32 | 9,9 | **9,85** | 9,86 | 4,04 | 3,98 | 4,09 | 0 | 0 | 0 | 9,61 | 9,65 | 9,89 |
| o2 | 8,33 | 8,11 | 8,43 | 9,89 | 9,88 | 9,88 | 4,04 | 4,04 | 4,04 | 0 | 0 | 0 | 9,77 | 9,86 | 9,59 |
| o3 | 8,11 | 8,35 | 8,48 | 9,91 | 9,89 | 9,86 | 3,95 | 4,07 | 4,04 | 0 | 0 | 0 | 9,96 | 9,87 | 9,94 |
| s1 | 0 | 0 | 0 | 10,77 | 10,28 | 10,53 | 6,09 | 6,09 | 6 | 9,61 | 9,77 | 9,96 | 0 | 0 | 0 |
| s2 | 0 | 0 | 0 | 10,51 | 10,52 | 10,53 | 6,02 | 6,05 | 6,02 | 9,64 | 9,86 | 9,87 | 0 | 0 | 0 |
| s3 | 0 | 0 | 0 | 10,53 | 10,55 | 10,49 | 6,09 | 6,02 | 6 | 9,9 | 9,6 | 9,94 | **0** | **0** | 0 |

Figure 4: Matrix of the amount of cross-traffic values between hosts in Mbps.

## 4.2 Software Equipment

For the experiments, the following software and technologies have been used:

1) **RMDT** — Reliable Multi-Destination Transport Protocol is a C++ software library, developed at the Future Internet Lab Anhalt at Anhalt University of Applied Sciences. It provides point to multipoint data transport functionality [1] using UDP. It uses **BQL** congestion control [9] which is tolerant of big delays and dramatic packet loss rates.

2) **Dataclone** — is the transfer application based on RMDT. In the experiments, it is configured to allocate 100 MB of RAM for both send and receive buffers. For the experiments, Dataclone has been set to the third CPU core. The most actual **v1.0.5** version has been used.

3) **Kite2** — is a software application, written in C++, developed at the Future Internet Lab Anhalt at Anhalt University of Applied Sciences. It is an implementation of the modified *AvB* active probing algorithm which can work in 10 Gbps links [4, 3].

4) **Cascade** — is a client-server application, written in *Python*, developed at the Future Internet Lab Anhalt at Anhalt University of Applied Sciences.. Cascade provides ALM functionality, it collects metrics such as RTT, using ICMP packets and AvB, using Kite2. Based on collected metrics it calculates transmission routes using MST or DCMST. In the client mode, it orchestrates Dataclones as transport software and communicates between other server instances.

5) **tc** — an open-source utility, which is used to configure Traffic Control in the Linux kernel. With this tool, the bandwidth of the interface is shaped down to **100 Mbps**. Traffic Control configured as follows:

```
qqdisc cbq 1: root refcnt 2 rate 10Mbit \
(bounded,isolated) prio no-transmit
qdisc sfq 30: parent 1:30 limit 127p quantum \
1514b depth 127 divisor 1024 perturb 10sec
qdisc sfq 10: parent 1:10 limit 127p quantum \
1514b depth 127 divisor 1024 perturb 10sec
qdisc sfq 20: parent 1:20 limit 127p quantum \
1514b depth 127 divisor 1024 perturb 10sec
qdisc ingress ffff: parent ffff:fff1 ----------------

class cbq 1: root rate 10Mbit (bounded,isolated) \
 prio no-transmit
class cbq 1:1 parent 1: rate 100Mbit (bounded,isolated) \
 prio 5
class cbq 1:10 parent 1:1 leaf 10: rate 100Mbit prio 1
class cbq 1:20 parent 1:1 leaf 20: rate 90Mbit prio 2
class cbq 1:30 parent 1:1 leaf 30: rate 80Mbit prio 2
```

6) **iperf** – is an open-source utility for performing network throughput measurements. For the experiments, iperf has been set to the second CPU core. The cross-traffic between hosts has been created with the following command:

```
// the process attached to core 2 with taskset 0x1
// the client started as
taskset 0x1 iperf -c DESTINATION_IP -b 10M -u -t 1000000
 // the server started as
taskset 0x1 iperf -s
```

The software testbed configuration is shown in Figure 5. **Cascade** represents the ALM system. It orchestrates the probing software, such as **Kite2** in case of AvB metric, or sends **ICMP** packets in case of RTT. It configures Dataclone as a sender, receiver, or relay node. It defines the source and destination for the data, which needs to be transmitted. The **iperf** utility is used as a cross-traffic generator. The **tc** util-ity restricts the datarates within 100 Mbps, to neglect the CPU and NIC as the bottleneck and to reduce the spendings for AWS traffic.

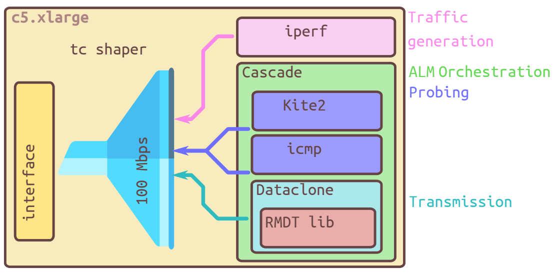


Figure 5: Software equipment scheme.

## 5 EXPERIMENTAL RESULTS

Experiments have been made in two scenarios: network with generated **cross-traffic**, described in section 4.1, and without interfering traffic, this scenario called **pure network**. Both RTT and AvB metrics have been tested in these conditions. Each test case has been repeated in 10 trials.

### 5.1 RTT Metric

In both cases, with and without cross-traffic, the **cascade** has built the same data distribution tree, which is shown in Figure 6. The tree constructed based on the metrics collected during the probing state of the **cascade** work. The delay matrix is shown in Figure 3. Cells with red bold values are related to paths built with RTT metrics.

The experiment without additional cross-traffic in the links given the transmission paths configuration provides **45.81** Mbps.

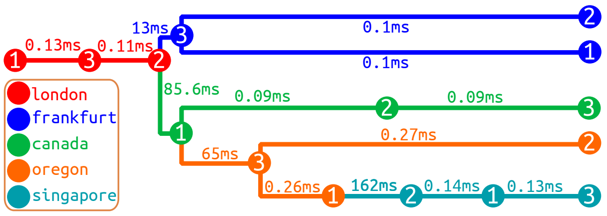


Figure 6: The tree generated by MST algorithm in the AWS network cloud infrastructure using RTT as a metric (in milliseconds).

In the presence of cross-traffic in the links, the same tree provides **34.89** Mbps. The cross-traffic generated for the test is shown in Figure 4. Cells with red bold values are related to paths built with RTT met-rics.

### 5.2 AvB Metric

In the case of links without additional traffic, AvB metric shows it's inefficiency. Since the links in the network are relatively pure from the interfering traffic, the AvB estimation probing returns similar met-rics, e.g 100 Mbps for all edges. This leads to uncertainty in the tree construction. The trees obtained during all 10 trials of an experiment with a "pure" network, were completely different and gained from 15 - 24 Mbps, the average data rate of the experiments is **23.65** Mpbs. The Figure 7 demonstrates one of the given trees.

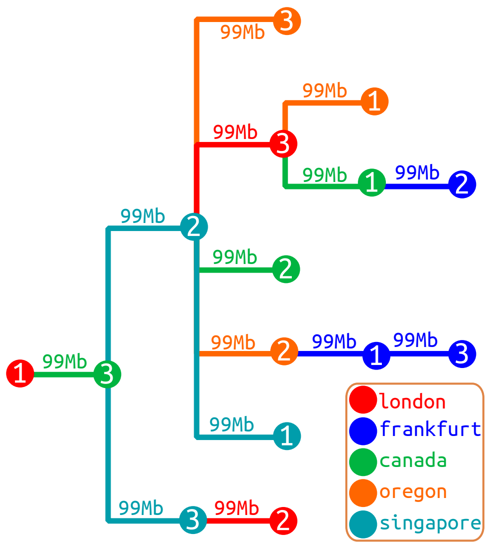


Figure 7: The tree generated by MST algorithm in the pure network cloud infrastructure using AvB as a metric (in Mbps).

However, in the presence of cross-traffic, shown in Figure 4, the edges of the tree, obtained with MST highlighted with red borders. The tree constructed us-ing the AvB metric is shown in Figure 8. The data rate achieved in the test is **44.65** Mbps, which is similar to the result of an experiment with the RTT metric with-out cross-traffic.

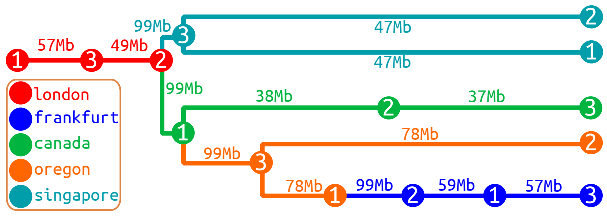


Figure 8: The tree generated by MST algorithm in the AWS network cloud infrastructure in the presence of cross-traffic using AvB as a metric (in Mbps).

## 6 CONCLUSIONS

The results of experiments allow to conclude the fol-lowing:

1) There are cases when available bandwidth as a metric for ALM can achieve higher performance for data transmission than using delay between hosts as a metric.

2) AvB gives insufficient results in the case of a ”pure” network.

3) Even in the modest presence of cross-traffic, the given transport system based on the RTT metric loses about 20 % of the data rate. This can be seen in the resulting Table 2.

4) For the AvB, as well as for delay metric it’s not necessary to be as precise as possible, it’s only necessary to establish the right relation between all edges of the network graph.

Table 2: Results of the data transmission experiments in Mbps.

| | AvB | RTT |
|---|---|---|
| Cross Traffic | 44.65 | 34.89 |
| Pure Network | 23.65 | 45.81 |

Since there are cases where one metrics is prefer-able than others to get the higher performance of the system, it makes sense to find criteria for the decision which metric should be used in the given situation. Finding such criteria is a preferable approach for the ongoing work. These criteria will be helpful in the process of developing the metric which will take into account both factors at once RTT and AvB.

## ACKNOWLEDGMENTS

This work has been funded by Volkswagen Foun-dation for trilateral partnership between scholars and scientists from Ukraine, Russia and Germany within the CloudBDT project: “Algorithms and Methods for Big Data Transport in Cloud Environments”.

## REFERENCES

[1] A. V. Bakharev, E. Siemens and V. P. Shuvalov, “Anal-ysis of performance issues in point-to-multipoint data transport for big data,” in 2014 12th International Con-ference on Actual Problems of Electronics Instrument Engineering (APEIE). IEEE, 2014, pp. 431-441.

[2] K. Karpov, D. Kachan, N. Mareev, V. Kirova, D. Syzov, Siemens and V. Shuvalov, “Adopting Minimum Spanning Tree Algorithm for Application-Layer Reli-able Mutlicast in Global Mutli-Gigabit Networks,” in Proceedings of the 7th International Conference on Ap-plied Innovations in IT, 2019.

[3] V. Kirova, E. Siemens, D. Kachan, O. Vasylenko and K. Karpov, “Optimization of Probe Train Size for Available Bandwidth Estimation in High-speed Net-works,” in MATEC Web of Conferences, vol. 208. EDP Sciences, 2018, p. 02001.

[4] V. Kirova, K. Karpov, E. Siemens, I. Zander, O. Va-sylenko, D. Kachan and S. Maksymov, “Impact of Modern Virtualization Methods on Timing Precision and Performance of High-Speed Applications,” Future Internet, vol. 11, no. 8, p. 179, 2019.

[5] M. Amad, A. Boudries and L. Badis, “Application Layer Multicast Based Services on Hierarchical Peer to Peer Architecture,” Applied Mechanics and Materials, vol. 892, pp. 64-71, June 2019.

[6] R. Muzaffar, E. Yanmaz, C. Raffelsberger, C. Bettstet-ter and A. Cavallaro, “Live multicast video stream-ing from drones: an experimental study,” Autonomous Robots, vol. 44, no. 1, pp. 75–91, January 2020.

[7] A. Sampaio and P. Sousa, “An adaptable and ISP-friendly multicast overlay network,” Peer-to-Peer Net-working and Applications, vol. 12, no. 4, pp. 809-829, July 2019.

[8] Q. Liu, R. Tang, H. Ren and Y. Pei, “Optimizing mul-ticast routing tree on application layer via an encoding-free non-dominated sorting genetic algorithm,” Applied Intelligence, vol. 50, no. 3, pp. 759-777, March 2020.

[9] N. Mareev, D. Kachan, K. Karpov, D. Syzov and E. Siemens, “Efficiency of BQL Congestion Control under High Bandwidth-Delay Product Network Condi-tions,” in Proceedings of the 7th International Confer-ence on Applied Innovations in IT, 2019.